Probabilistic Networks for Verifying Automated Testing of High Speed Telecommunication Equipment through the Development Lifecycle

R. Sterritt, K. Adamson, E. P. Curran, C.M. Shapcott
University of Ulster at Jordanstown
Shore Road, Newtownabbey, BT37 0QB, Northern Ireland

Abstract

Learning probabilistic networks for expert system applications has seen a great surge of research activity in recent years. This paper reports on the adoption of probabilistic networks for verifying the pass/fail result of automated testing at the software verification stage in the development lifecycle of high-speed telecommunications equipment.

The focus is on learning Bayesian Belief Networks (BBNs) from automated test data on a per test execution basis. This facilitates result classification and assurance, taking advantage of their graphical nature to provide accountability of the decision.

Keywords: Bayesian Belief Networks, automated testing, intelligent systems, telecommunication networks.

1 Introduction

Within Nortel Networks' NITEC (Northern Ireland Telecommunications Engineering Centre) R&D lab, high capacity broadband transmission and switching equipment are designed and developed. This complex mix of hardware, software and firmware must conform to international standards to facilitate heterogeneous global networks.

During the development cycle for each release of a product a significant proportion of the time is taken up with testing, commonly estimated at 60%. As the product becomes larger and more complex the ability to comprehensively test and verify the operation within the decreasing timeframe to market becomes increasingly difficult. Automation offers the potential to decrease this overhead.

Traditional manual testing of telecommunications equipment was expensive in terms of time spent, costs involved and even de-motivation of specialised engineers due to the repetitive task. Automation offered a competitive advantage in terms of reduced cost, reduced time to market, enhanced quality and "freeing-up" of specialised engineers for further investigating and solving of problems areas discovered from the testing [27].

The disadvantage to automation is that the experimental approach to testing is lost [21],[27]. A test script will not spot anomalous behaviour that an engineer would have.

Automation offers a rich data trail which can then be utilised to compensate for the loss in live experimentation by the engineer. Hidden in that data should be indications that any anomalies have occurred. Each execution of an individual test leaves behind a statistical 'footprint' which can be presented graphically.

The assumption is that the footprints can be utilised for a wider-based identification (classification) of a pass or fail of an individual test. In any case where there is a sufficiently large number of pass and fail footprints available, it should be possible to use classification techniques, such as a neural network [22], to generate a pass/fail classifier for automated testing. Yet a drawback of many classification techniques, including neural networks, is that they do not provide any explanation of the decision.

Inducing a probabilistic network from the data provides a much more visual footprint. Probabilistic networks [1], [17], in which relationships between variables can be represented by the existence of links between them, have an intuitive appeal. They are easy to "read" if represented graphically and can summarise fairly complex relationships succinctly.

0-7803-6583-6/00/$10.00 © 2000 IEEE

2 Automated Testing

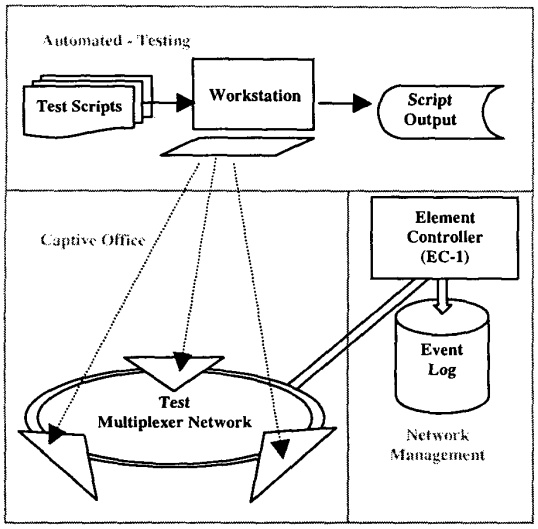

Figure 1: Verification Automated Testing of Network Elements (NE)

Within NITEC, during the testing phase of releases 6/7 TN-1X (STM-1) multiplexer software, a significant amount of knowledge and data have been collected from a fixed topology. However, analysing the data collected has proved to be extremely difficult due to the complexity of the interactions and the large quantity of data involved. The required testing to achieve the necessary coverage is increasing significantly with network complexity. NITEC engineers have committed substantial effort in an attempt to automate this testing. These automated test scripts provide substantial cost and time savings over manual testing, being faster to perform as well as enabling test runs overnight and at weekends. Yet they cannot provide the same coverage. An expert engineer in attendance can spot anomalies a test script could not. For instance, incorrect intermediate states which are generated during an automatic test but which are resolved by the end of the test are not easily detected. In order for the engineers to spot these states, they need to analyse all data produced during the test. To do this manually would negate the time advantage gained by the partial automation [25]. These rare incorrect intermediate states may be indicating problems which exist in the software that is under development but yet the test may have been considered a pass [26].

In addition, the expertise of interpreting data produced from this testing is not always available. It is important that this knowledge be readily available if testing and subsequent analysis is going to be fully automated. Justifying decisions on judging tests as pass/fails is very important as well as being able to make suggestions as to where to look for the root cause of problems. A means of reducing the cognitive overload on the test engineers is essential due to the unabated growth in complexity of the networks.

3 Probabilistic Network Induction

There are few cases where the researcher has advance knowledge of the structure of the probabilistic network. Therefore there is a need to induce or learn the structure from the data which has the form of frequencies of the different variable values occurring in combination.

Extensive research has been carried out on the induction of probabilistic networks. Buntine [7] and Heckerman [17] provide good surveys of the problem. Aliferis and Cooper [1], Breese [2], Breese et al.[3], Bouckaert [4], Buntine [5],[6], Chickering et al. [8], Cooper and Herskovits [10],[11], Dempster [12], Goldman and Charniak [15], and Geiger and Heckerman [13],[18],[19],[20] are just some of the literature which considers the induction problem.

Unfortunately the general problem is NP-hard [8]. For a given number of variables there is a very large number of potential graphical structures which can be induced. To determine the best structure one should, in theory, fit the data to each possible graphical structure, score the structure, and then select the structure with the best score. Consequently algorithms for learning networks from data are usually heuristic, once the number of variables gets to be of reasonable size. There are $2^{k(k-1)/2}$ distinct possible independence graphs for a k-dimensional random vector: this translates to 64 probabilistic models for k= 4, and 32,768 models for k = 6.

In this work, several different induction algorithms were developed and tested using telecommunications data supplied by NITEC.

3.1 The Chow and Liu algorithm

In this algorithm, due to Chow and Liu [9] the mutual information between pairs of variables is calculated and those variables with the highest value are connected. The algorithm continues with successive elimination of variables. It has the advantage of simplicity but generates only tree structures. The mutual information between two variables, a and b is defined as:

$$\mu_{ab} = \sum_{i,j} \Pr(a=i, b=j) \log \frac{\Pr(a=i, b=j)}{\Pr(a=i)\Pr(b=j)} \qquad (1)$$

It can be shown that if the independence graph is a tree (that is if there are no cycles in the I-map) the best fit to the data is obtained by the Chow and Liu algorithm.

The main advantage of using tree-structured graphs is that they are simple to compute. The probability tables can be calculated directly from the marginal counts of the variables - maximum likelihood estimators are obtained by simply computing the pairwise marginal counts of the variables.

3.2 CAEGA

The Cause and Effect Genetic Algorithm (CAEGA) developed by Sterritt et al. [23],[24] target good networks which they 'breed' in order to produce new, potentially better nets. It was found to give good solutions and was selected as an innovative algorithm for extracting BBNs from telecommunications data.

The algorithm takes as input a modified form of the Bayesian Belief Network Interchange Format proposed by the UAI community [29]. In this format each variable is specified with its name, a description and the number of values that it can have, and a descriptor for each possible value. In the data table each tuple is replaced with a vector of integer values, one integer for each value, and the number of occurrences of each distinct vector also appears. This format considerably compresses the storage requirement for the input file. The genetic algorithm also accepts a user-specified list of net structures (defined by child-parent lists) which may have been generated by other algorithms such as that developed by Chow and Liu.

The algorithm works in the following way. Internally, breeders are specified by a square matrix of binary values in which a unit value indicates a parent-child relationship between two variables and a zero indicates that there is no direct relationship. An initial breeding stock is created from the breeders specified in the input data file and by random creation of graphs, which represent the net structure. Because the graph must be acyclic not all binary matrices define valid graphs, and any invalid matrices created by breeding or mutation are pruned until they are valid. At each generation, pairs of breeders are selected randomly to exchange genetic material and thereby create new breeders. Others are selected for random mutations. Scoring was done using the posterior probability function in Cooper and Herskovits [11]. Because there is a problem of overfitting with these models – the saturated model in which one node is picked arbitrarily and taken as a child of all the others is a perfect match to the data – thus displaying a need for a penalty to be applied to the score. In our implementation the penalty effect is obtained by limiting the number of parents each child may possess. The algorithms all have output which includes belief network specifications in standard format.

The following section discusses the application of learning these probabilistic networks for testing assurance.

4 Probabilistic Networks for Testing Assurance

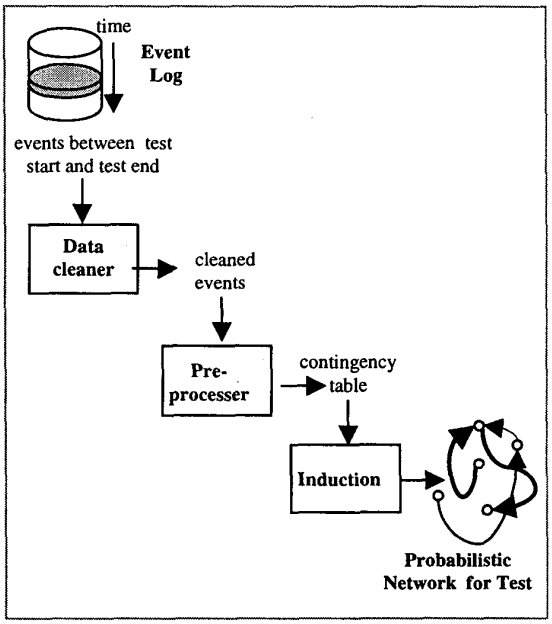

Figure 2: Inducing probabilistic networks from test data for test assurance

The process, in common with most machine learning approaches, uses a relation-based approach to the data. Information concerning the behaviour of test cases is assumed to be available and is passed, after suitable pre-processing, to an induction process, which extracts a model - in our case a Bayesian Belief Network (BBN) - from the data.

The induction process attempts to optimise the network – in the sense that it searches for the network structure that best fits the data. The BBN extracted would normally be used as the basis for an expert system. In this case it provides a model for classifying the test.

The realisation of the process can be viewed in Figure 2, containing a Data Cleaner, Pre-processor and Induction component. It provides an abstracted model of the

behaviour in the form of a BBN. This is achieved by examining the events that occur during a test run from the event log which resides on the network controller.

The hypothesis is that the BBN expresses the behaviour that has occurred during the test to either provide assurance that the test "pass" is a correct assumption or to highlight the behavioural anomalies.

Due to the nature of the Synchronous Digital Hierarchy (SDH) system where a fault can produce a cascade of alarms reported back to the network controller, large quantities of verbose data are generated. In one particular experiment a single fault produced 6Mb of binary data.

Data cleaning and data pre-processing is often an understated process. Hatonen et al. [16] found it to require 80% of the knowledge discovery process in some domains.

Test assurance and confidence levels will rely on the ability of cross-referencing the extracted behaviour for that test with previous behaviour for the same test on a different test run or a similar test. This necessitates the storage of BBN for each specific test run.

In manual processing terms the tests produce a large amount of data, yet in data mining terms the induction of a BBN on a per test-script basis is from a relatively small amount of data. As such it was found that CAEGA which was designed as a BBN data mining algorithm was not suitable.

It did not produce a relatively constant BBN to allow comparisons between tests. This is not surprising for several reasons; firstly genetic algorithms (GAs) have a random element to assist in the prevention of being caught at a local maximum. Secondly, GAs are designed to perform in a large search space and an under-sized population affects their behaviour by causing premature convergence [14]. The data for a single script is too small to effectively use. There are possible methods to work around this problem and one approach is to adapt the GA to perform competition at the family level instead of globally. It is believed that this approach is less prone to error with undersized populations [28], yet the random element to the GA could still cause differences in the BBN due to the algorithm and not the test data.

The adaptation of OMI (Optimisation of Mutual Information) was used as an alternative for this situation. As already mentioned the algorithm was one of the first in use for the induction of Bayesian networks, published by Chow and Liu [9]. Basically, the algorithm attempts to find a structure in which the strengths of the edges between the pairs of nodes are maximised. The edge strength is measured by the mutual information between the nodes. The implementation used was a greedy algorithm in which edges are added in order of strength while preserving an acyclic structure. Initial experimentation with the adapted OMI algorithm indicates a more suitable algorithm. This is explained by the fact that each time a variation occurs in the induced BBN it is as a result of the test, not the algorithm.

5 A Motivating Example

The following example study shows the results from two separate runs of a sample auto-test script. Both runs were considered to have passed by the script. The hardware and software configuration had not changed significantly between runs.

The script performs the simulation of simple faults into a test network via the command line user interface (CLUI) on the element controller. The network consists of two multiplexers named Enfield and Acton. The faults were induced on Enfield. The sample commands shown demonstrate the disconnection of tributary port 1 (in slot 2 of the multiplexer) then its reconnection after a time period. In the sample test ports 2 - 8 were also disconnected then after a time period reconnected in the same way.

```
Cmd=c/n/d S6-k111&S7-k111 S2-
1,User=clui1
Cmd=c/n/c S7-1-J1-K111&S6-1-J1-K111
S2-1,User=clui1
Cmd=c/n/d S6-k112&S7-k112 S2-
2,User=clui1
Cmd=c/n/c S7-1-J1-K112&S6-1-J1-K112
S2-2,User=clui1
Cmd=c/n/d S6-k113&S7-k113 S2-
3,User=clui1
Cmd=c/n/c S7-1-J1-K113&S6-1-J1-K113
S2-3,User=clui1
Cmd=c/n/d S6-k121&S7-k121 S2-
4,User=clui1
Cmd=c/n/c S7-1-J1-K121&S6-1-J1-K121
S2-4,User=clui1
Cmd=c/n/d S6-k122&S7-k122 S2-
5,User=clui1
Cmd=c/n/c S7-1-J1-K122&S6-1-J1-K122
S2-5,User=clui1
Cmd=c/n/d S6-k123&S7-k123 S2-
6,User=clui1
Cmd=c/n/c S7-1-J1-K123&S6-1-J1-K123
S2-6,User=clui1
Cmd=c/n/d S6-k131&S7-k131 S2-
7,User=clui1
```

```
Cmd=c/n/c S7-1-J1-K131&S6-1-J1-K131
S2-7,User=clui1
Cmd=c/n/d S6-k132&S7-k132 S2-
8,User=clui1
Cmd=c/n/c S7-1-J1-K132&S6-1-J1-K132
S2-8,User=clui1
```

Thus in total 16 commands were performed (8 sets of disconnection and reconnections). Table 1 displays a breakdown of the event types that were recorded in the event log on the element controller during these runs. Note no other activity was occurring on the network during the experiment.

Table 1. Breakdown of recorded events during experiment

Event Type	Run 1	Run 2
Alarm Events	476	463
Login Events	106	106
User Action Events	16	16
Message Tool Events	159	160
System Error Events	1	1
Total number of events	758	746

The auto-test script consisted of 16 actual commands (recorded as user action events) but also required 106 login actions throughout the script. Although over 400 alarm instances occurred in both runs, only 5 actual alarm types transpired, which are shown in Table 2.

Table 2. Alarm types that occurred during the experiment

Alarm Event Type	Explanation
PPI-AIS	PDH Physical Interface - Alarm Indication Signal
PPI-Unexp_Signal	PDH Physical Interface - Unexpected Signal
LP-PLM	Lower order Path - Path Label Mismatch
INT-TU-LOP	Internal - Tributary Unit – Loss of Pointer
INT-TU-AIS	Internal - Tributary Unit - Alarm Indication Signal

The induction of a BBN from each set of data produced differing results (Table 3 and Table 4) that indicated an anomaly. Since both tests were recorded as passes, these differences gave the indication of the need for investigation. The differences originate from the addition of an alarm in the second run, INT-TU-LOP, and the occurrence of only 9 INT-TU-AISs as opposed to 15.

Table 3. BBN Induction results for auto-test run 1

Frequencies of Alarm Occurrence
0, PPI-AIS, 192
1, INT-TU-LOP, 0
2, PPI-Unexp_Signal, 8
3, LP-PLM, 8
4, INT-TU-AIS, 15

Strength of Edges (mutual information score)
0 >:2, 0.121383
2 >:3, 0.113237
0 >:3, 0.0832668
0 > 4, 0.0676941
3 > 4, 0.052712

Probabilistic Connections
p(PPI_Unexp_Signal | PPI_AIS)
p(LP_PLM | PPI_AIS, PPI_Unexp_Signal)
p(INT_TU_AIS | PPI_AIS, LP_PLM)

Table 4. BBN Induction results for auto-test run 2

Frequencies of Alarm Occurrence
0, PPI-AIS, 192
1, INT-TU-LOP, 1
2, PPI-Unexp_Signal, 8
3, LP-PLM, 8
4, INT-TU-AIS, 9

Strength of Edges (mutual information score)
0 > 2, 0.123822
2 > 3, 0.103088
0 > 3, 0.0957039
0 > 4, 0.0637544
3 > 4, 0.0629661
2 > 4, 0.028953
1 > 3, 0.00722244
1 > 2, 0.00667615
0 > 1, 0.00467317

Probabilistic Connections
p(PPI_AIS)
p(INT_TU_LOP | PPI_AIS, PPI_Unexp_Signal, LP_PLM)
p(PPI_Unexp_Signal | PPI_AIS)
p(LP_PLM | PPI_AIS, PPI_Unexp_Signal)
p(INT_TU_AIS | PPI_AIS, PPI_Unexp_Signal, LP_PLM)

Upon consultation with engineers it was discovered that "the occasional occurrence of INT-TU-LOP during break/make connections was a characteristic of the TN-1X product".

Also INT-TU-LOP is a major alarm while INT-TU-AIS is a minor one. LOP is directly above AIS in the masking hierarchy. The reduced number of INT-TU-AIS alarms is most likely due to the masking effect of INT-TU-LOP.

Therefore in this case it was decided that the differences in the 'footprints' (Figure 3 and Figure 4) should not indicate a fail. Note that the thickness of the edges in these figures represents the strength of the connections between variables in the belief network. The strengths of the additional edges in Figure 4 appear so low that visually there is not a strong

difference between the footprints. It is relatively simple to set an ignore threshold above this for classification purposes.

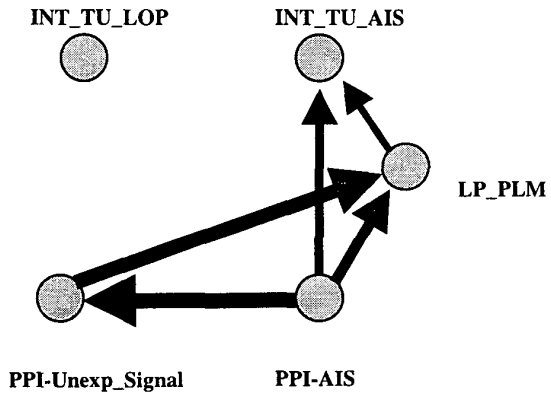

Figure 3: BBN (footprint) of Test 1

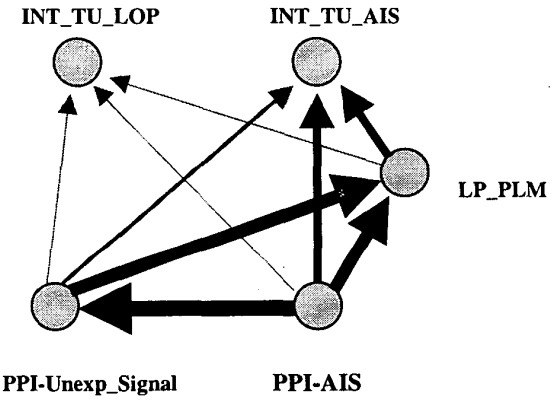

Figure 4: BBN (footprint) of Test 2

6 Conclusion and Future Direction

This paper has reported on the adoption of probabilistic networks for verifying the pass/fail result of automated testing at the software verification stage in the development lifecycle of high-speed telecommunications equipment.

Automated testing offers advantages such as cost reductions in terms of extensive overtime requirements, being able to test and repeatability test again and again in a fraction of the manual testing time, improve the job quality of test engineers freeing them up to examine and solve the real problems encountered. In doing so the quality and reliability of the products is increased.

Yet the disadvantage of automation is that the experimental approach to testing is lost. The engineer is no longer at hand to spot anomalies and investigate.

Inducing a probabilistic network from the data offers a means to summarise the behaviour of the network during the test. Comparing the results to an expected norm or previous test results offers a means of classifying the result and thus providing assurance of the automated result.

Since probabilistic networks are easily readable when represented graphically and summarise fairly complex relationships succinctly, they provide an explanation of the decision unlike other AI approaches.

The approach introduced in this paper, offers great promise. From initial experimentation it would appear that the BBNs can be used as a classification technique. They cover all events that have occurred during a test and therefore provide the means to make up for the lack of a test engineer at the scene monitoring for anomalous activity.

At present it is generally valid to compare a test run against the previous nights test run. Yet the environment is not constant. Throughout the development lifecycle, from initial software beta releases to final product, the behaviour of the software gradually changes and as such the induced network will be different throughout the lifecycle. Future work would involve storing the network with other information as a case in a case-based reasoning (CBR) system. This further enhancement will offer an automated adaptive decision making capability to the process.

Acknowledgements

We acknowledge funding for this work by IRTU Start programme, research project ITS 07: the GARNET project, 1997-1999 with industrial collaboration from Northern Ireland Telecommunications Engineering Centre (NITEC), Nortel Networks.

References

[1] C. Aliferis, G. Cooper, "An Evaluation of an Algorithm for Inductive Learning of Bayesian Belief Networks Using Simulated Data Sets", *Proc. Of the 10th Conf. On Uncertainty in Artificial Intelligence*, pp.8-14, 1994.

[2] J. S. Breese, "Construction of Belief and Decision Networks", *Computational Intelligence*, Vol. 8(4), pp.624-647, 1992.

[3] J. S. Breese, R. P. Goldman, M. P. Wellman, "Introduction to the Special Section of Knowledge-based Construction of Probabilistic Networks and Decision Models", *IEEE Trans. On Systems, Man and Cybernetics*, Vol. 24(11), pp.1577-1579, 1994.

[4] R. R. Bouckaert, "Properties of Bayesian Belief Network Learning Algorithms", *Proc. of the 10th Conf. On Uncertainty in Artificial Intelligence*, 1994.

[5] W. Buntine, "Operations for Learning with Graphical Models", *J. of Artificial Intelligence Research*, Vol. 2, pp.159-225, 1994.

[6] W. Buntine, "Chain Graphs for Learning", *Proc. of the 11th Annual Conf. On Uncertainty in Artificial Intelligence*, 1995.

[7] W. Buntine, "A Guide to the Literature on Learning Probabilistic Networks from Data", *IEEE Transactions on Knowledge and Data Engineering*, Vol. 8, No. 2, pp.195-210, 1996.

[8] D. M. Chickering, D. Geiger, D. Heckerman, "Learning Bayesian networks is NP-hard". *Technical Report MSR-TR-94-17, Microsoft Research*, 1994.

[9] C. J. K. Chow, C. N. Liu., "Approximating discrete probability distributions *with dependence trees", IEEE Trans. Information Theory,* Vol. 14(3), pp.462-467, 1968.

[10] G. F. Cooper, E. Herskovits, "A Bayesian Method for Constructing Bayesian Belief Networks from Databases", *Proc. of the 7th Ann. Conf. On Uncertainty in Artificial Intelligence*, 1991.

[11] G. F. Cooper, E. Herskovits, "A Bayesian Method for the Induction of Probabilistic Networks from Data", *Machine Learning*, Vol. 9, pp.309-347, 1992.

[12] A. P. Dempster, "Construction and Local Computation Aspects of Network Belief Functions", R. M. Oliver, J. Q. Smith, eds., Influence Diagrams, Belief Nets and Decision Analysis, pp.121-124, Wiley & Sons, 1990.

[13] D. Geiger, D. Heckerman, "Learning Bayesian Networks: A Unification for Discrete and Gaussian Domains", *Proc. 11th Annual Conference on Uncertainty and Artificial Intelligence*, 1995.

[14] D. E. Goldberg, "Sizing Populations for Serial and Parallel Genetic Algorithms", *Proc. 3rd Int. Conf. On Genetic Algorithms,* pp.70-79, 1989.

[15] R. P. Goldman, E. Charniak, "A Language for Construction of Belief Networks", IEEE Trans. On Pattern Analysis and Machine Intelligence, Vol. 15(3), pp.196-208, 1993.

[16] K. Hatonen, M. Klemettinen, H. Mannila, P. Ronkainen, H. Toivionen, "Knowledge Discovery from Telecommunication Network Alarm Databases", *12th ICDE,* 1996.

[17] D. Heckerman, "A Tutorial on Learning Bayesian Networks", *Technical Report, MSR-TR-96-06 Microsoft Research*, 1995.

[18] D. Heckerman, D. Geiger, "Learning Bayesian Networks", Technical Report MSR-TR-95-02, *Microsoft Research*, 1995.

[19] D. Heckerman, D. Geiger, D. M. Chickering, "Learning Bayesian Networks: The Combination of Knowledge and Statistical Data", *Proc. 10th Annual Conference on Uncertainty and Artificial Intelligence,* pp.293-301, 1994.

[20] D. Heckerman, "Bayesian Networks for Knowledge Discovery" In U.M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, R. Uthurusamy (Eds.), *Advances in Knowledge Discovery and Data Mining* AAAI Press / The MIT Press, pp.273-305, 1996.

[21] I. D. Hicks, G. J. South, A. O. Oshisanwo, "Automated Testing as an Aid to Systems Integration", *BT Tech J.*, Vol. 15 (3), pp.26-36, 1997. *Data Analysis*, Vol. 19, pp.155-175, 1995.

[22] D. E. Rumelhart, B. Widrow, M. A. Lehr, "The Basic Ideas in Neural Networks", *Methods of the Brain and Evolution, Journal of Commucations of the ACM*, Vol. 37(3), pp.87-92, 1994.

[23] R. Sterritt, K. Adamson, C. M. Shapcott, D. A. Bell, F. McErlean, "Using A.I. For The Analysis Of Complex Systems", *Proc. IASTED Int. Conf. Artificial Intelligence and Soft Computing*, pp.113-116, 1997.

[24] R. Sterritt, K. Adamson, M. Shapcott, N. Wells, D. A. Bell, W. Liu, "P-CAEGA: A Parallel Genetic Algorithm For Cause And Effect Networks", *Proc. IASTED Int. Conf. Artificial Intelligence and Soft Computing*, pp.105-108, 1997.

[25] R. Sterritt, K. Adamson, C. M. Shapcott, E. P. Curran, "Adapting An Architecture For Knowledge Discovery In Complex Telecommunication Systems For Testing Assurance" *Proceedings of the NIMES 98 Conference on Complex Systems, Intelligent Systems and Interfaces,* pp.37-39, 1998.

[26] R. Sterritt, E. P. Curran, K. Adamson, C. M. Shapcott, "Application Of AI For Automated Testing In Complex Telecommunication Systems", *EXPERSYS 98 - Artificial Intelligent Applications,* Eds. D.A. Jacobs, IITT-International, pp.97-102, 1998.

[27] R. Sterritt, K. Adamson, E. P. Curran, C. M. Shapcott, "Towards Intelligent Automated Testing In The Development Lifecycle Of High Speed Telecommunication Equipment", *Proceedings of the International Conference On Artificial Intelligence (IC-AI)*, 2000.

[28] D. Thierens, D. Goldberg, "Elitist Recombination: an integrated selection recombination GA", *Proc. 1st IEEE Con. On Evolutionary Computation*, pp.508-512, 1994.

[29] Decision Theory Group, "Proposal for a Bayesian Network Interchange Format", http://www.research.microsoft.com/research/dtg/bnformat/proposal.htm, 1996.